What is claimed is:

1. A computerized method comprising, in each of a plurality of processors including a first processor and a second processor:

(a) loading a first vector register with addressing values;

(b) loading a second vector register with operand values;

(c) determining which, if any, element addresses of the first vector register have a value that duplicates a value in another element address; and

(d) selectively adding certain elements of the second vector of operand values based on the element addresses the duplicated values;

(e) loading, using indirect addressing from the first vector register, elements from memory into a third vector register;

(f) adding values from the third vector register and the second vector of operand values to generate a result vector; and

(g) storing the result vector to memory using indirect addressing.

2. The method of claim 1, wherein the set of operations (a), (b), (c), and (d) is performed substantially in parallel in the plurality of processors, and the set of operations (e), (f), and (g) is performed serially, one processor at a time.

3. The method of claim 1, further comprising

executing an ordered Msync operation before the set of operations (e), (f), and (g); and

executing an end ordered Msync operation after the set of operations (e), (f), and (g).

4. The method of claim 3, wherein the set of operations (a), (b), (c), and (d) is performed substantially in parallel in the plurality of processors.

5. The method of claim 1, further comprising

executing a first barrier synchronization operation before the set of operations

(e), (f), and (g) in all of the plurality of processors;

executing a second barrier synchronization operation before the set of operations (e), (f), and (g) in the second processor;

executing the set of operations (e), (f), and (g) in the first processor and then executing a second barrier synchronization operation in the first processor to satisfy the second barrier synchronization in the second processor, and executing a third barrier synchronization in the first processor; and

executing the set of operations (e), (f), and (g) in the second processor and then executing a third barrier synchronization operation in the second processor to satisfy the third barrier synchronization in the first processor.

6. The method of claim 5, wherein the set of operations (a), (b), (c), and (d) is performed substantially in parallel in the plurality of processors.

7. The method of claim 1,

wherein the determining of duplicates includes:

generating each respective address value for a sequence of addressed locations within a constrained area of memory containing 2^N consecutive addresses using an N-bit value derived from each respective addressing value of the first vector register,

generating each respective data value of a first sequence of values by combining at least a portion of each respective addressing value of the first vector register to a respective one of a sequence of integer numbers,

storing the first sequence of values to the constrained memory area using the generated sequence of respective address values,

loading a second first sequence of values from the constrained memory area using the generated sequence of respective address values, and

comparing the first sequence of values to the second sequence of values; and

wherein the loading of the third vector register includes loading elements

from locations specified by addressing values corresponding to indications of positive compares from the comparing;

wherein addresses of the elements from memory are calculated by adding each respective addressing value to a base address;

wherein the adding includes a floating-point addition operation that produces at least one element of the result vector as an ordered-operation floating point summation of an element of the loaded third vector register and a plurality of respective elements of the original second vector of operand values corresponding to elements of the first vector of addressing values having identical values, and

wherein for the storing of the result vector of elements to memory, elements are stored to locations specified by addressing values corresponding to indications of positive compares.

8. A computerized method comprising:

(a) within a first vector processor:

loading a first vector register in the first vector processor with addressing values;

loading a second vector register in the first vector processor with operand values;

determining which, if any, element addresses of the first vector register in the first vector processor have a value that duplicates a value in another element address;

selectively adding certain elements of the second vector of operand values in the first vector processor based on the element addresses the duplicated values;

(b) within a second vector processor:

loading a first vector register in the second vector processor with addressing values;

loading a second vector register in the second vector processor with operand values;

determining which, if any, element addresses of the first vector register in the second vector processor have a value that duplicates a value in another element address;

selectively operating on certain elements of the second vector of operand values in the second vector processor based on the element addresses the duplicated values;

(c) performing a synchronization operation that ensures that prior store operations effectively complete in at least the second vector processors before the following (d) operations;

(d) within the first vector processor:

loading, using indirect addressing from the first vector register, elements from memory into a third vector register in the first vector processor;

operating on values from the third vector register and the second vector of operand values in the first vector processor to generate a first result vector; and

storing the first result vector to memory using indirect addressing;

(e) performing a synchronization operation that ensures that the storing of the first result vector effectively completes before the following (f) operations; and

(f) within the second vector processor:

loading, using indirect addressing from the first vector register, elements from memory into a third vector register in the second vector processor;

operating on values from the third vector register and the second vector of operand values in the second vector processor to generate a second result vector; and

storing the second result vector to memory using indirect addressing.

9. The method of claim 8, wherein each of the operating on functions includes adding.

10. The method of claim 9, wherein the adding includes a floating-point addition operation that produces at least one element of the result vector as an ordered-operation floating point summation of an element of the loaded third vector register and a plurality of respective elements of the original second vector of operand values corresponding to elements of the first vector of addressing values having identical values.

11. The method of claim 8,

wherein the determining of duplicates includes:

generating each respective address value for a sequence of addressed locations within a constrained area of memory containing 2^N consecutive addresses using an N-bit value derived from each respective addressing value of the first vector register,

generating each respective data value of a first sequence of values by combining at least a portion of each respective addressing value of the first vector register to a respective one of a sequence of integer numbers,

storing the first sequence of values to the constrained memory area using the generated sequence of respective address values,

loading a second first sequence of values from the constrained memory area using the generated sequence of respective address values, and

comparing the first sequence of values to the second sequence of values.

12. The method of claim 8, further wherein the loading of the third vector register of each processor includes loading elements from locations specified by addressing values corresponding to indications of positive compares from the comparing operation.

13. The method of claim 8, further wherein indirect addresses of the elements from memory are calculated by adding each respective addressing value to a base

address.

14. A system comprising:

a first vector register having addressing values;

a second vector register having operand values;

circuitry programmed to determine which, if any, element addresses of the first vector register have a value that duplicates a value in another element address;

circuitry programmed to selectively add certain elements of the second vector of operand values based on the element addresses the duplicated values;

circuitry programmed to load, using indirect addressing from the first vector register, elements from memory into a third vector register;

circuitry programmed to add values from the third vector register and the second vector of operand values to generate a result vector; and

circuitry programmed to store the result vector to memory using indirect addressing.

15. The system of claim 14,

wherein the circuitry programmed to determine duplicates includes:

circuitry programmed to generate each respective address value for a sequence of addressed locations within a constrained area of memory containing 2^N consecutive addresses using an N-bit value derived from each respective addressing value of the first vector register,

circuitry programmed to generate each respective data value of a first sequence of values by combining at least a portion of each respective addressing value of the first vector register to a respective one of a sequence of integer numbers,

circuitry programmed to store the first sequence of values to the constrained memory area using the generated sequence of respective address values,

circuitry programmed to load a second sequence of values from the constrained memory area using the generated sequence of respective

address values, and

circuitry programmed to compare the first sequence of values to the second sequence of values; and

wherein the circuitry programmed to load the third vector register loads elements from locations specified by addressing values corresponding to indications of positive compares;

wherein addresses of the elements from memory are calculated by adding each respective addressing value to a base address;

wherein the circuitry programmed to add includes a floating-point adder that produces at least one element of the result vector as an ordered-operation floating point summation of an element of the loaded third vector register and a plurality of respective elements of the original second vector of operand values corresponding to elements of the first vector of addressing values having identical values.

16. The system of claim 14, further comprising:

circuitry programmed to perform the set of operations (a), (b), (c), and (d) substantially in parallel in the plurality of processors, and

circuitry programmed to perform the set of operations (e), (f), and (g) serially, one processor at a time.

17. The system of claim 14, further comprising:

circuitry programmed to execute an ordered Msync operation before the set of operations (e), (f), and (g); and

circuitry programmed to execute an end ordered Msync operation after the set of operations (e), (f), and (g).

18. The system of claim 17, further comprising:

circuitry programmed to perform the set of operations (a), (b), (c), and (d) substantially in parallel in the plurality of processors.

19. The system of claim 14, further comprising:

circuitry programmed to execute a first barrier synchronization operation before the set of operations (e), (f), and (g) in all of the plurality of processors;

circuitry programmed to execute a second barrier synchronization operation before the set of operations (e), (f), and (g) in the second processor;

circuitry programmed to execute the set of operations (e), (f), and (g) in the first processor and then executing a second barrier synchronization operation in the first processor to satisfy the second barrier synchronization in the second processor, and executing a third barrier synchronization in the first processor; and

circuitry programmed to execute the set of operations (e), (f), and (g) in the second processor and then executing a third barrier synchronization operation in the second processor to satisfy the third barrier synchronization in the first processor.

20. The system of claim 19, further comprising circuitry programmed to perform the set of operations (a), (b), (c), and (d) substantially in parallel in the plurality of processors.

21. A system comprising:

(a) a first vector processor that includes:

means for loading a first vector register in the first vector processor with addressing values;

means for loading a second vector register in the first vector processor with operand values;

means for determining which, if any, element addresses of the first vector register in the first vector processor have a value that duplicates a value in another element address;

means for selectively adding certain elements of the second vector of operand values in the first vector processor based on the element addresses the duplicated values;

(b) a second vector processor that includes:

means for loading a first vector register in the second vector processor with addressing values;

means for loading a second vector register in the second vector processor with operand values;

means for determining which, if any, element addresses of the first vector register in the second vector processor have a value that duplicates a value in another element address;

means for selectively operating on certain elements of the second vector of operand values in the second vector processor based on the element addresses the duplicated values;

(c) means for performing a synchronization operation that ensures that prior store operations effectively complete in at least the second vector processors before the operations of the following (d) means;

(d) within the first vector processor:

means for loading, using indirect addressing from the first vector register, elements from memory into a third vector register in the first vector processor;

means for operating on values from the third vector register and the second vector of operand values in the first vector processor to generate a first result vector; and

means for storing the first result vector to memory using indirect addressing;

(e) performing a synchronization operation that ensures that the storing of the first result vector effectively completes before the operations of the following (f) means; and

(f) within the second vector processor:

means for loading, using indirect addressing from the first vector register, elements from memory into a third vector register in the second vector processor;

means for operating on values from the third vector register and the second vector of operand values in the second vector processor to generate a second result vector; and

means for storing the second result vector to memory using indirect

addressing.

22. The system of claim 21, wherein each of the means for operating on functions includes an adder.

23. The system of claim 22, further

wherein the adder includes a floating-point adder that produces at least one element of the result vector as an ordered-operation floating point summation of an element of the loaded third vector register and a plurality of respective elements of the original second vector of operand values corresponding to elements of the first vector of addressing values having identical values

24. The system of claim 23, wherein the means for determining of duplicates further includes:

means for generating each respective address value for a sequence of addressed locations within a constrained area of memory containing 2^N consecutive addresses using an N-bit value derived from each respective addressing value of the first vector register,

means for generating each respective data value of a first sequence of values by combining at least a portion of each respective addressing value of the first vector register to a respective one of a sequence of integer numbers,

means for storing the first sequence of values to the constrained memory area using the generated sequence of respective address values,

means for loading a second first sequence of values from the constrained memory area using the generated sequence of respective address values, and

means for comparing the first sequence of values to the second sequence of values.

25. The system of claim 21, further wherein the means for loading of the third

vector register of each processor includes means for loading elements from locations specified by addressing values corresponding to indications of positive compares from the comparing operation.

26. The system of claim 21, further wherein indirect addresses of the elements from memory are calculated by adding each respective addressing value to a base address.

27. A computer-readable medium having instructions stored thereon for causing a suitably programmed information-processing system to execute a method comprising:

loading a first vector register with addressing values;

loading a second vector register with operand values;

determining which, if any, element addresses of the first vector register have a value that duplicates a value in another element address;

selectively adding certain elements of the second vector of operand values based on the element addresses the duplicated values;

loading, using indirect addressing from the first vector register, elements from memory into a third vector register;

adding values from the third vector register and the second vector of operand values to generate a result vector; and

storing the result vector to memory using indirect addressing.